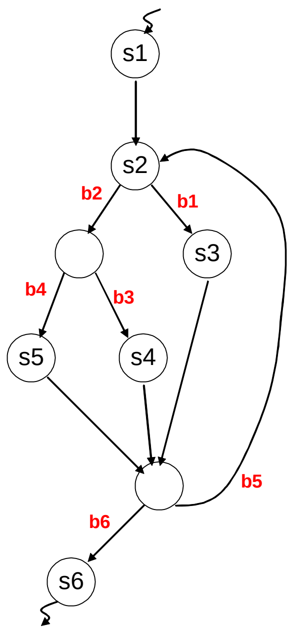
Problem Set 2 Solution Notes

1. a)

```
BINARY(list,N,key:IN;
    found,mid:OUT)

s1 {  lo := 1
      hi := N
      found := FALSE
      REPEAT
s2      mid := (lo + hi) div 2
            C1:
        IF key=list[mid] THEN
s3        found := TRUE
        ELSE
                C2:
          IF key<list[mid] THEN
s4          hi := mid-1
          ELSE
s5          lo := mid+1
          END-IF-ELSE
        END-IF-ELSE
              C3:         C4:
      UNTIL (found OR lo>hi)
                   TF, FT, FF
s6    return(found, mid)
```

Control-Flow Graph:

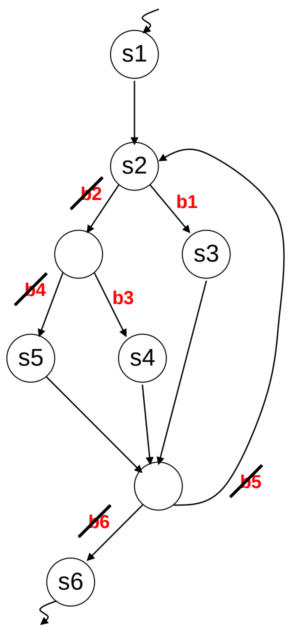


b) result_1: false val_1: 5
Path traversed: <s1, s2, s5, s2, s5, s2, s5, s6>
Coverage: None

```
BINARY(list,N,key:IN;
    found,mid:OUT)

~~s1~~ {  lo := 1
          hi := N
          found := FALSE
          REPEAT
~~s2~~      mid := (lo + hi) div 2
                C1: FFF
            IF key=list[mid] THEN
s3            found := TRUE
            ELSE
                    C2: FFF
              IF key<list[mid] THEN
s4              hi := mid-1
              ELSE
~~s5~~          lo := mid+1
              END-IF-ELSE
            END-IF-ELSE
                  C3: FFF     C4: FFT
          UNTIL (found OR lo>hi)
                       TF, ~~FT~~, ~~FF~~
~~s6~~    return(found, mid)
```

Control-Flow Graph:

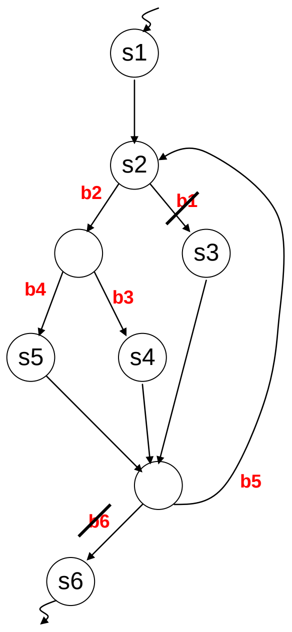


c) result_2: true   val_2: 3
Path traversed: <s1, s2, s3, s6>
Coverage: None

**BINARY(list,N,key:IN;
found,mid:OUT)**

Control-Flow Graph:

```
      { lo := 1
~~s1~~  { hi := N
      { found := FALSE
        REPEAT
~~s2~~      mid := (lo + hi) div 2
               C1: T
            IF key=list[mid] THEN
~~s3~~        found := TRUE
            ELSE
                 C2:
              IF key<list[mid] THEN
s4              hi := mid-1
              ELSE
s5              lo := mid+1
              END-IF-ELSE
            END-IF-ELSE
                  C3: T         C4: F
        UNTIL (found OR lo>hi)
                  ~~TF~~, FT, FF
~~s6~~  return(found, mid)
```

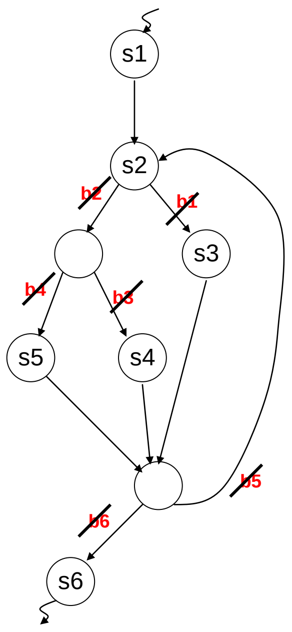


d) result_3: true   val_3: 2
Path traversed: <s1, s2, s4, s2, s5, s2, s3, s6>
Coverage: statement, branch, condition, feasible compound condition

**BINARY(list,N,key:IN;
found,mid:OUT)**

Control-Flow Graph:

```
      { lo := 1
~~s1~~  { hi := N
      { found := FALSE
        REPEAT
~~s2~~      mid := (lo + hi) div 2
               C1: FFT
            IF key=list[mid] THEN
~~s3~~        found := TRUE
            ELSE
                 C2: TF
              IF key<list[mid] THEN
~~s4~~          hi := mid-1
              ELSE
~~s5~~          lo := mid+1
              END-IF-ELSE
            END-IF-ELSE
                  C3: FFT       C4: FFF
        UNTIL (found OR lo>hi)
                  ~~TF~~, FT, ~~FF~~
~~s6~~  return(found, mid)
```



2. Two counterexamples are required. Consider the pseudocode program:

```
if A OR B then
    S1
else
    S2
end_if_then_else

if C AND D then
    S3
else
    S4
end_if_then_else
```

Counterexample #1:

	A	B	C	D	path
1	T	T	T	T	TT
2	T	F	T	F	TF
3	F	T	F	T	TF
4	F	F	F	F	FF

These test cases provide compound condition coverage but not path coverage.

Therefore, compound condition coverage does not subsume path coverage.

Counterexample #2:

	A	B	C	D	path
5	T	F	T	T	TT
6	T	F	F	T	TF
7	F	F	T	T	FT
8	F	F	F	T	FF

These test cases provide path coverage but not compound condition coverage.

Therefore, path coverage does not subsume compound condition coverage.

Together, the two counter-examples above show that compound condition coverage and path coverage are independent.